Examen RC **2002-2003**
Fiifo5

Le barême précisé est indicatif. La clarté de la rédaction sera pris en compte dans la note.
Le sujet comprend 4 pages.

1. Profondeur itérative et recherche non informée (4 pts)

a. Soit le graphe d'états de la figure 1. Les successeurs d'un noeud sont développés dans l'ordre alphabétique. Appliquez une recherche en largeur d'abord, en profondeur d'abord sur cet arbre (numérotez l'ordre de développement des noeuds). Donnez les propriétés de ces deux types de recherche ainsi que leur complexité en temps et en espace.

b. Appliquez l'algorithme de profondeur itérative donné en annexe à ce même arbre de recherche en prenant f(n) = profondeur de n dans l'arbre de recherche, pour tout noeud n appartenant à l'arbre de recherche. Quel est l'avantage de la recherche à profondeur itérative par rapport à la recherche en profondeur d'abord et largeur d'abord ?

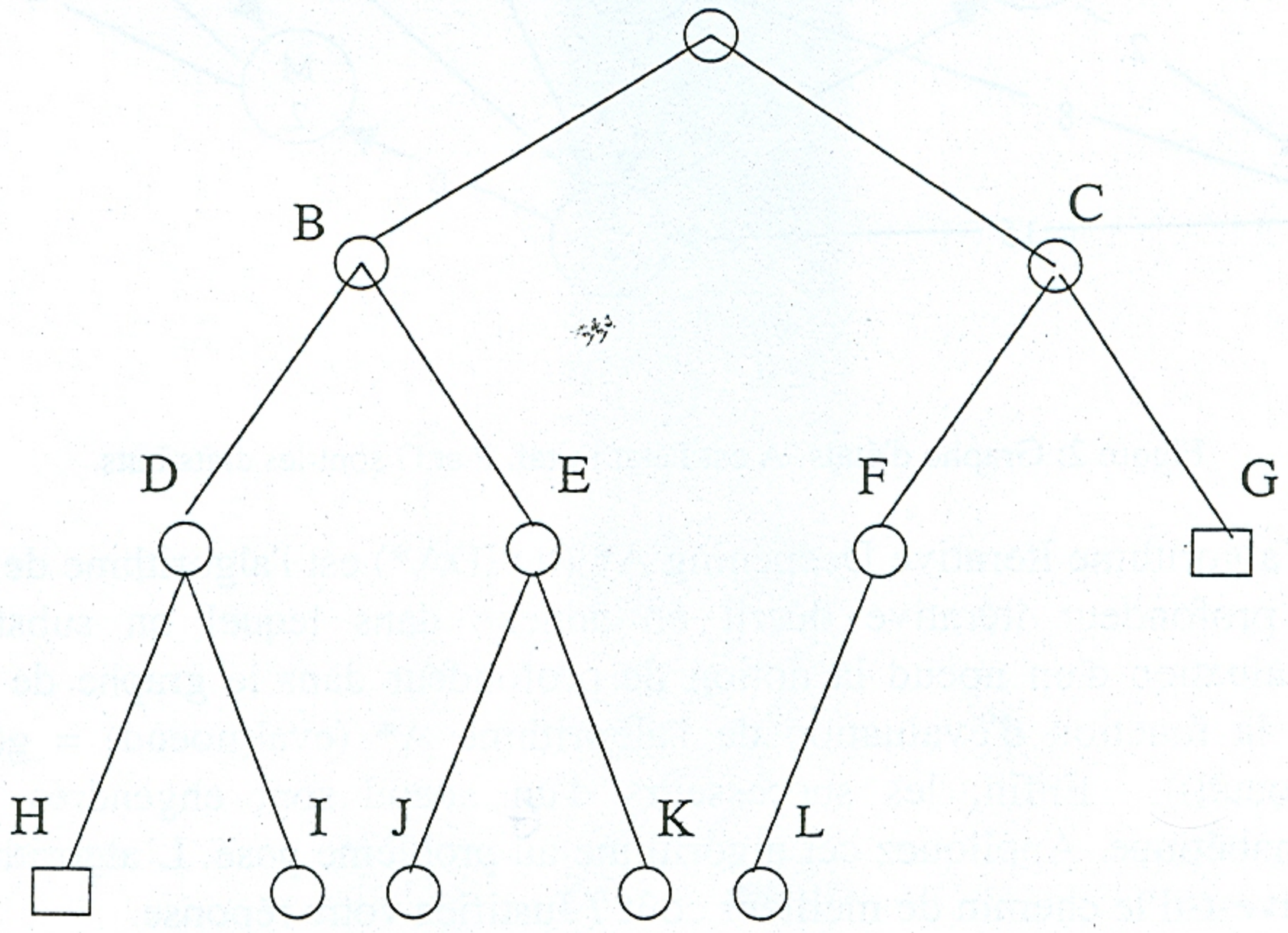

Fig 1: Graphe d'état. Les deux noeuds solutions sont notés par des carrés

2. Recherche A* et profondeur itérative (10 pts)

Le graphe d'états de la figure 2 décrit un ensemble d'états A,B,C,... reliés entre eux par des arcs valués. Le problème posé consiste à trouver le meilleur chemin (en terme de coût) entre l'état initial A et un des deux états terminaux P ou Q. A chaque nœud n est associé une valeur numérique h(n) caractérisant de façon heuristique une approximation de la distance séparant n et un des états buts.

a. Appliquez l'algorithme A* afin de trouver un chemin entre A et un des états buts. Ce chemin est-il le chemin de meilleur coût ? Justifiez votre réponse. Quel est le nombre de nœuds engendrés lors de la recherche ? Quel est le nombre de nœuds développés ?

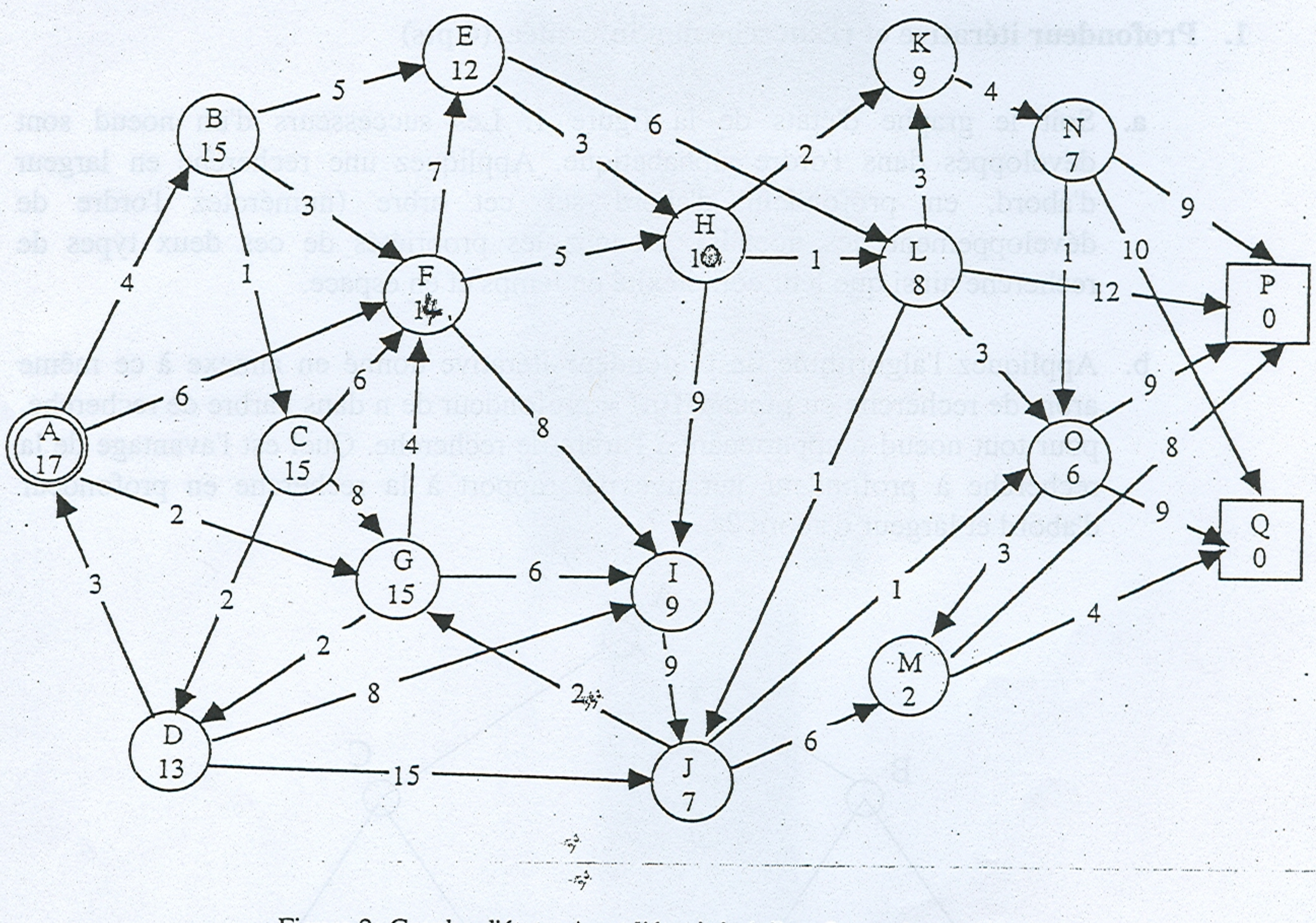

Figure 2: Graphe d'états, A est l'état inital, P et Q sont les états buts.

b. L'algorithme Iterative Deepening A* (ou IDA*) est l'algorithme de recherche en profondeur itérative décrit en annexe dans lequel on substitue pour l'évaluation d'un noeud la notion de profondeur dans le graphe de recherche par la fonction d'évaluation de l'algorithme A* (eval(noeud) = g(noeud) + h(noeud)). Enfin, les successeurs d'un noeud sont engendrés par ordre alphabétique. Appliquez cet algorithme au problème posé. L'algorithme IDA* trouve-t-il le chemin de meilleur coût ? Justifiez votre réponse.

c. Comparez le nombre de nœuds développés au cours de toute la recherche IDA* par rapport à la recherche A*. Quand et pourquoi la stratégie IDA* vous paraît-elle plus adaptée que l'algorithme A* ?

3. Recherche Alpha béta (6 pts)

Considérons l'arbre de jeu donné à la figure 3. Appliquez l'algorithme alpha-béta de gauche à droite (ordre alphabétique sur les successeurs) puis de droite à gauche (ordre anti-alphabétique sur les successeurs). Que constatez vous ? Expliquez.

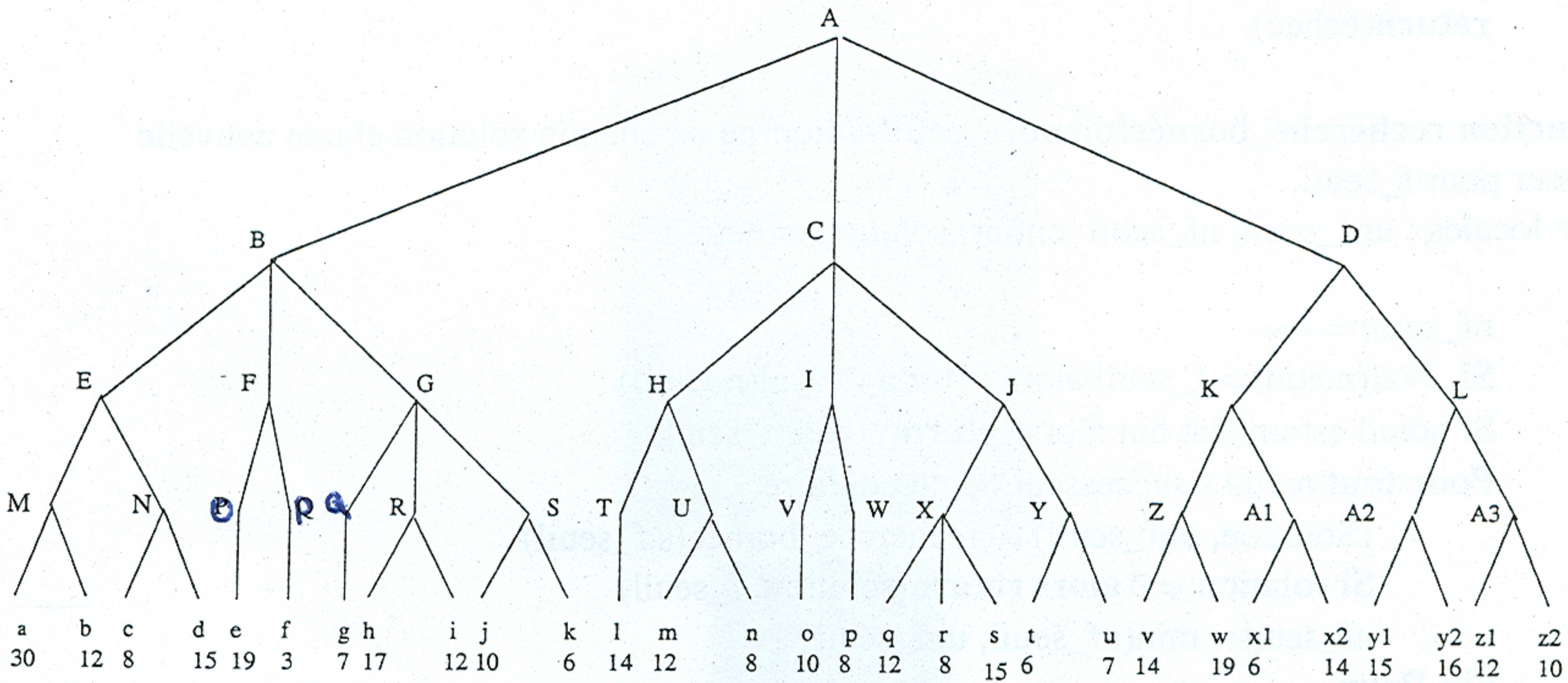

Fig 3: Arbre de jeu. La racine est max.

Annexe

I. Algorithme de recherche en profondeur itérative

Fonction profondeur_itérative(problème) retourne chemin_solution

racine ← état_initial
f_seuil ← éval(état_initial)
tant que f_seuil ≠ + ∝ faire
(solution, f_seuil) ← recherche_bornée(racine,f_seuil)
si solution ≠∅ alors **return**(solution)
fin tant que
return(échec)

Fonction recherche_bornée(nœud, f_seuil) retourne un chemin solution et une nouvelle valeur pour f_seuil.
Var locales : unf_seuil, nf_seuil: entier; solution : chemin.

nf_seuil = + ∝
Si éval(nœud) > f_seuil alors **return**(∅,éval(noeud))
Si nœud est un état but alors **return**(nœud, f_seuil)
Pour tout nœud s successeur de nœud faire
(Solution, unf_seuil) ← recherche_bornée(s,f_seuil)
Si solution ≠∅ **alors return**(solution, f_seuil)
nf_seuil← min(nf_seuil, unf_seuil)
Fin Pour
return(∅, nf_seuil).

II. Indications :

1. Un des plus courts chemins pour le problème de la figure 2 est A-F-H-J-O-M-Q et son coût est de 21.
2. La première itération de IDA* sur le problème de la fig. 2 est donnée à la figure suivante:

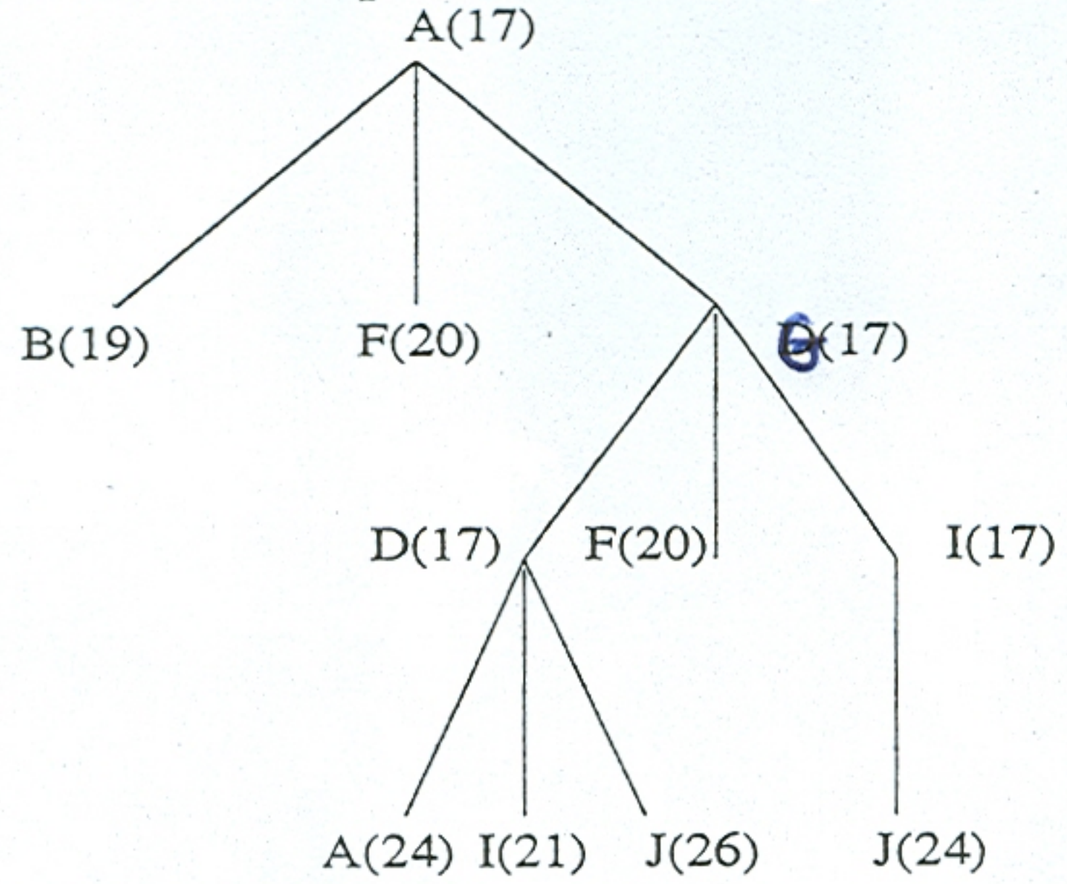

Fig 4 : Une itération de l'algorithme de recherche en profondeur d'abord

Les nombres notés entre parenthèses sont les valeur f des noeuds. La valeur f_seuil retournée à la fonction profondeur_itérative après la première itération est 19, la plus petite valeur de f dans l'arbre ayant mené à un échec de développement.